APPLICATION FOR UNITED STATES LETTERS PATENT

For

A METHOD AND SYSTEM FOR COLLECTING PERFORMANCE RELATED INFORMATION FOR A REMOTE DEVICE

Inventors:

Konstantinos Roussos
Neelesh Madhukar Thakur
Zdenko Kukavica

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8300

Attorney's Docket No.: 005693.P046
Client Docket No.: P01-1756

"Express Mail" mailing label number: EV336589936US
Date of Deposit: 3-16-04
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
KATHLEEN K. MUTO
(Typed or printed name of person mailing paper or fee)
Kathleen K. Muto
(Signature of person mailing paper or fee)
3-16-04
(Date signed)

A METHOD AND SYSTEM FOR COLLECTING PERFORMANCE RELATED INFORMATION FOR A REMOTE DEVICE

FIELD OF THE INVENTION

[0001] At least one embodiment of the invention relates to data storage, and in particular to the collecting of performance information relating to a storage device.

BACKGROUND

[0002] As used herein, the term "storage device," refers to any physical host device capable of performing storage related functions. Thus, storage devices include, but are not limited to storage servers (filers), network attached storage (NAS) devices, direct attached storage (DAS) devices, etc. Typically, the storage devices are coupled within a storage network or storage system.

[0003] A performance object refers to a logical or physical subsystem of a storage device. For example, the performance objects for a storage device may include disks, volumes, Central Processing Units (CPUs), Host Bus Adaptors (HBAs), Logical Unit Numbers (LUNs), etc.

[0004] Typically, a storage device has a number of performance counters to collect performance data relating to each of its performance objects. Examples of performance data include statistical data, system status data, system configuration data, etc. The data collected by the counters is useful in assisting a storage system administrator to identify performance related problems in a storage network. However, often a storage system administrator has no

knowledge of the counters available for a storage device, and may further lack the skill to collect data from the counters. Typically, if a storage system administrator finds that there is a storage related problem with a storage system, the administrator contacts a customer support center and obtains a script from the customer support center to collect and view counter data for the storage devices in the storage network, in an attempt to diagnose the cause of the performance related problem. Such an approach is not always successful, since the actual historical data relating to when the performance related problem actually occurred may not have been collected, and may, therefore, not be available.

SUMMARY OF THE INVENTION

[0005] In one embodiment, the invention provides a method comprising for each predefined tuple containing a counter and an object; and for at least one remote device in a network that has the particular object and counter contained in the tuple, automatically sampling the counter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] **Figure 1** shows a high level block diagram of a network architecture, within which embodiments of the present invention may be practiced;

[0007] **Figure 2** shows a representation of each of the hosts/filers of Figure 1 in terms of its logical and physical subsystems;

[0008] **Figure 3** shows a high level block diagram of the components of a management module installed within the management server;

[0009] **Figure 4** shows a mapping table that may be used to practice embodiments of the invention; and

[0010] **Figure 5** shows a high level block diagram of hardware that may be used to implement any of the hosts, clients, and the management server, shown in Figure 1.

DETAILED DESCRIPTION

[0011] Embodiments of the present invention relate to a management tool that automatically discovers performance objects, instances of the performance objects, and counters associated with said instances for each remote device/host in a network, and automatically creates instantiations of counters based on a predefined counter group template comprising mappings of performance objects to counters.

[0012] The management tool keeps track of all instances of performance objects so that new counters can be automatically instantiated in the case of new instances of performance objects being added, and instantiated counters can be automatically deleted in the case of an existing instance of a performance object being removed.

[0013] Advantageously, the management tool allows the network administrator to collect performance data for an arbitrary number of performance objects, without having *a priori* knowledge of what counters are available for each performance object, and without having to run a specialized script to collect the performance data. Other advantages of embodiments of the invention will be apparent from the detailed description below.

[0014] Referring now to Figure 1 of the drawings, reference numeral 100 generally indicates a network architecture, within which embodiments of the present invention may be practiced. As will be seen, the network architecture 100 includes a number of remote devices/ hosts 102. The components of each host 102, are described with reference to Figure 5 of the drawings, below. In one

embodiment, each host 102 may be a filer supplied by Network Appliance, Inc. Each host 102 is coupled to a management server 104 via a communications path 106. The communications path 106 may include a transport layer, which supports a transport protocol, for example, the Hypertext Transfer Protocol (HTTP), or the User Datagram Protocol (UDP). Information sent over the transport protocol may be encoded using a variety of protocols, for example, the Simple Network Management Protocol (SNMP), or the Extensible Markup Language (XML).

[0015] The management server 104 may have an architecture in accordance with that shown in Figure 5 of the drawings. The purpose of the management server 104 is to provide storage management functions to a number of client devices (clients) 108. In order to provide these storage management functions, the management server runs an application 104.1, known as a management module (MM). In one embodiment, the MM may be the DataFabric® product supplied by Network Appliance, Inc. The management server 104 also includes a database 104.2, and an operating system 104.3. In one embodiment, the operating system 104.3 may be, for example, the Solaris, Window, or Linux operating system.

[0016] In one embodiment, the clients 108 may have an architecture such as that shown in Figure 5 of the drawings. Each of the clients includes a client application 108.1, which, in use, invokes one or more Application Program Interfaces (APIs) in order to display performance data collected by the MM application 104.1 as will be described in greater detail below.

[0017] Each client 108 is coupled to the management server 104 via a communications link 110. For example, in one embodiment the communications link 100 supports a Remote Procedure Call (RPC) made over a suitable transport protocol, e.g., UDP or HTTP.

[0018] Figure 2 of the drawings shows a representation of each of the hosts/filers 102 in terms of its logical and physical subsystems. As will be seen, these logical and physical subsystems may include disks 200, disk volumes 202, Central Processing Units (CPUs) 204, network interfaces 206, Host Bus Adaptors (HBAs) 208, Logical Unit Numbers (LUNs) 210, a Network File System (NFS) processing block 212, a Common Internet File System (CIFS) processing block 214, a Fibre Channel Protocol (FCP) processing block 216, and an Internet Small Computer System Interface (iSCSI) processing block 218. Each of the logical and physical subsystems of the host 102 described above defines a performance object from which performance data may be collected, using one or more performance counters. Typically, the data collected by the performance counters may be used by a storage network administrator to identify problems in the storage network. Examples of counters include counters to measure CPU load, the number of CIFS operations, the number of FCP operations, the number of iSCSI operations, etc.

[0019] Referring now to Figure 3 of the drawings, there is shown a high level block diagram of the components of an MM 300, which is representative of one embodiment of the MM 104.1 of Figure 1. As will be seen, the MM 300 includes

a discovery thread or block 302, a counter setup block 304, a data collection block 306, and a data presentation block 308.

[0020] The discovery block 302 automatically discovers and reports to the MM 300 (a) all performance objects on the various hosts/filers 102 within the network architecture 100, (b) all instances of the performance objects, and (c) all counters associated with said instances. In one embodiment, in order to perform the above-described steps (a) to (c), each host 102 includes a set of Application Program Interfaces (APIs) to discover and report the objects, instances and counters to the MM 300. In one embodiment, the MM 300 may poll or query each host 102 to obtain the information on the performance objects, instances of the performance counters, and counters associated with the instance, using the APIs. In one embodiment, the APIs may include an agent to read a configuration file for each host in order to obtain information about the performance objects, instances, and counters that form part of the host and to report this information to the MM 300. In one embodiment, the agent may be configured to automatically report changes in the performance objects for a given host to the MM 300 whenever they occur. For example, if a new instance of a performance object is added to a host then this event is automatically reported to the MM 300. Alternatively, if an existing instance of a performance object is removed from a host, then this event is also reported to the MM 300. In one embodiment, the agent may be configured using SNMP traps.

[0021] Referring now to Figure 3A of the drawings it will be seen that the counter setup block 304 includes a thread 304.1 to instantiate counters based on

the information provided by the discovery thread 302 on the performance objects associated with a particular host. In order to instantiate the counters, the thread 304.1 uses predefined mappings (tuples) of performance objects . These predefined mappings are called counter group templates. The counter group templates are stored in the database 104.2. An example of a counter group template is shown in Figure 4 of the drawings, which contains a table that maps objects (e.g., CPU, disks, etc.) to counter names (e.g., CPU_busy, disk_kb_written, and disk_kb_read). In this example, the counter CPU_busy measures the load on a particular CPU, the counter name disk_kb_written provides an indication of the number of kilobytes written to a disk, whereas the counter disk_kb_read provides an indication of the number of kilobytes read from a particular disk. A counter group template may also specify a sample buffer size, a sample period, and name for the counter group template. The sample buffer size specifies how long to keep sampled counter values in stable storage. The sample period specifies a period at which an instantiated counter is to be sampled. Shown below is an example of an algorithm, in pseudocode, that may be used to instantiate the counters:

[0022] <u>Algorithm to instantiate counters</u>:

for each (object, counter) in counter group template;

for each (object, instance, counter) in filer;

if object in filer matches object in counter group template; and counter in filer matches counter in database,

then instantiate counter and insert into list.

[0023] An instantiated counter is also known as a fully qualified counter since information on which counter of which instance of which object, of which filer/host is completely known. Thus, the output of instantiating the counters at block 304.1 is a fully qualified counter which has the path:

filer(host)/object/instance/counter name.

[0024] Referring to Figure 3, at block 304.2, the counter setup block 304 creates and/or updates counter groups. A counter group is a group of instantiated or fully-qualified counters. In one embodiment, a counter group includes the following elements: (1) one or more fully qualified counters, (2) a sample period at which the fully qualified counter is to be sampled, (3) a counter group name, and (4) one or more file names into which data sampled from the fully qualified counter is to be written. A counter group may also include a sample buffer size that specifies how long to keep counter values in stable storage.

[0025] Referring now to Figure 3B, there is shown a flowchart of operations performed by the data collection block 306 in order to collect data for each of the counters set up in the counter setup block 304. Referring to Figure 3B, at 306.1 for each counter group, the MM 300 reads the counter group information. At block 306.2, the MM 300 queries the filers for counter values based on the counter group information. In one embodiment, the MM 300 maintains a priority queue in which a number of data collection threads are arranged in a heap sorted by time. When the time for executing each thread arrives, the MM 300 de-

queues the thread from the heap and places the thread in a work queue. The MM 300 periodically de-queues each thread from the work queue and executes the thread. As a result of executing the threads in the work queue, performance data is obtained from the filers and inserted into the file referenced by the counter group. At block 306.3, the MM application writes the results of the queries into the file referenced by the counter group.

[0026] In one embodiment, the data collected by the data collection block 306 is stored in one or more flat files. In some cases, the data collected from the counters may be stored on a per filer/host basis. Thus, referring to Figure 1, all counter group data collected from filer 1 is stored separately from all counter group data collected from filer 2. Under this file storage scheme, if, for example, filer 2 has failed and is non-operational for some reason, then the collection of data from filer 1 is unaffected. Further, under this file storage scheme, advantageously, data collection can proceed even though a counter group is being updated or changed. Thus, data collection and order of discovery of the performance objects can occur in parallel.

[0027] Figure 3C of the drawings illustrates the components of the data presentation block 308 of the MM 300, in accordance with one embodiment. Referring to Figure 3C, it will be seen that data presentation block 308 includes a view configuration block 308.1. The view configuration block 308.1 receives input from a user selecting a predefined view associated with one or more counter groups. In response, the view configuration block 308.1 retrieves the predefined view from the database 104.2 and sends the performance data for the

counter group and the selected predefined view to the client, where the performance data is presented to a system administrator in accordance with the predefined view. For example, in one embodiment, the performance data and the predefined view may be sent as a markup language document. In another embodiment, the client may request the predefined view from the MM 300. After receiving the predefined view, the client retrieves data for the counters in the predefined view at a given rate set by the client.

[0028] Referring to Figure 5 of the drawings, reference numeral 500 generally indicates hardware that may be used to implement the hosts 102, the clients 108, or the management server 104. The hardware 500 typically includes at least one processor 502 coupled to a memory 504. The processor 502 may represent one or more processors (e.g., microprocessors), and the memory 504 may represent random access memory (RAM) devices comprising a main storage of the hardware 500, as well as any supplemental levels of memory e.g., cache memories, non-volatile or back-up memories (e.g. programmable or flash memories), read-only memories, etc. In addition, the memory 504 may be considered to include memory storage physically located elsewhere in the hardware 500, e.g. any cache memory in the processor 502, as well as any storage capacity used as a virtual memory, e.g., as stored on a mass storage device 510.

[0029] The hardware 500 also typically receives a number of inputs and outputs for communicating information externally. For interface with a user or operator, the hardware 500 may include one or more user input devices 506

(e.g., a keyboard, a mouse, etc.) and a display 508 (e.g., a Cathode Ray Tube (CRT) monitor, a Liquid Crystal Display (LCD) panel).

[0030] For additional storage, the hardware 300 may also include one or more mass storage devices 510, e.g., a floppy or other removable disk drive, a hard disk drive, a Direct Access Storage Device (DASD), an optical drive (e.g. a Compact Disk (CD) drive, a Digital Versatile Disk (DVD) drive, etc.) and/or a tape drive, among others. Furthermore, the hardware 500 may include an interface with one or more networks 512 (e.g., a local area network (LAN), a wide area network (WAN), a wireless network, and/or the Internet among others) to permit the communication of information with other computers coupled to the networks. It should be appreciated that the hardware 500 typically includes suitable analog and/or digital interfaces between the processor 502 and each of the components 504, 506, 508 and 512 as is well known in the art.

[0031] The hardware 500 operates under the control of an operating system 514, e.g., the operating system 104.3, and executes various computer software applications 516 (e.g., the MM 104.1), components, programs, objects, modules, etc. (e.g. a program or module which performs operations described above). Moreover, various applications, components, programs, objects, etc. may also execute on one or more processors in another computer coupled to the hardware 500 via a network 512, e.g. in a distributed computing environment, whereby the processing required to implement the functions of a computer program may be allocated to multiple computers over a network.

[0032] In general, the routines executed to implement the embodiments of the invention, may be implemented as part of an operating system or a specific application, component, program, object, module or sequence of instructions referred to as "computer programs." The computer programs typically comprise one or more instructions set at various times in various memory and storage devices in a computer, and that, when read and executed by one or more processors in a computer, cause the computer to perform operations necessary to execute elements involving the various aspects of the invention. Moreover, while the invention has been described in the context of fully functioning computers and computer systems, those skilled in the art will appreciate that the various embodiments of the invention are capable of being distributed as a program product in a variety of forms, and that the invention applies equally regardless of the particular type of signal bearing media (computer-readable media) used to actually effect the distribution. Examples of signal bearing media include but are not limited to recordable type media such as volatile and non-volatile memory devices, floppy and other removable disks, hard disk drives, optical disks (e.g., Compact Disk Read-Only Memory (CD ROMS), Digital Versatile Disks, (DVDs), etc.), among others, and transmission type media such as digital and analog communication links.

[0033] Although embodiments of the invention have been described with respect to storage devices, it should be borne in mind that those embodiments apply in general to any processing system or computer. Thus, for example, the MM application described above may be used to collect performance information

from any remote device/host that supports the APIs to report information on performance objects, instances of the performance objects, and counters associated with the instances in accordance with the above described techniques.